

Graph Splicing System

L. Jeganathan

and

R. Rama*

Department of Mathematics,
Indian Institute of Technology Madras,
Chennai, India

Abstract

The string splicing was introduced by Tom Head which stands as an abstract model for the DNA recombination under the influence of restriction enzymes. The complex chemical process of three dimensional molecules in three dimensional space can be modeled using graphs. The graph splicing systems which were studied so far, can only be applied to a particular type of graphs which could be interpreted as linear or circular graphs. In this paper, we take a different and a novel approach to splice two graphs and introduce a splicing system for graphs that can be applied to all types of graphs. Splicing two graphs can be thought of as a new operation, among the graphs, that generates many new graphs from the given two graphs. Taking a different line of thinking, some of the graph theoretical results of the splicing are studied.

1 Introduction

To understand and analyze well the complex structure as well as the evolutionary process of genes, researchers have long been searching for syntactical models. One such model was a grammatical model provided by formal language theory [1]. Yet, the grammar types in the Chomsky hierarchy was inadequate in describing the biological systems [2].

In his pioneering work, Tom Head has proposed an operation called ‘Splicing’ for describing the recombinant behavior of double-stranded DNA molecules [3] which established a new relationship between formal language theory and the study of informational macromolecules. Splicing operation is a formal model of the recombinant behavior of DNA molecules under the influence of restriction enzymes and ligases. Informally, splicing two strings means to cut them at points specified by the given substrings (corresponding to patterns recognized by restriction enzymes) and to concatenate the obtained fragments crosswise

*Corresponding author. Email : ramar@iitm.ac.in

(this corresponds to the ligation reaction). Since then, the theory of splicing has become an interesting area of formal language theory, where results of splicing systems on string languages (splicing systems were later renamed as H-systems to indicate the originator) gave new insights in some (closure) properties of families of string languages [4]. The mathematical study of the splicing operation on the strings has been investigated exhaustively, which lead to a language generating device viz., Extended H-systems (EH-systems) using the ‘splicing operation’ as the basic ingredient [5]. Several control mechanisms were suggested in increasing the computing power of EH systems with finite components, equivalent to the power of Turing machines. Thus, splicing operation on strings has lead to universal computing device (programmable DNA Computers based on splicing).

A splicing operation contains splicing rules of the form $(u_1, u_2; u_3, u_4)$, where u_1, u_2, u_3, u_4 are strings over some alphabet V . We apply the splicing rule to two strings $x_1 u_1 u_2 x_2, y_1 u_3 u_4 y_2$, (x_1, x_2, y_1, y_2 are strings over V^*). As a result, the new strings $x_1 u_1 u_4 y_2$ and $y_1 u_3 u_2 x_2$ are obtained. We use the modified definition of splicing as it appears in [4]

DNA sequences are three dimensional objects in a three-dimensional space. Some problems arise when they are described by one-dimensional strings. So, the other models of splicing were explored. In [7, 8, 9, 10], array splicing systems were studied. In [6],[15] graph splicing systems were discussed. But these systems cannot be applied to the graphs that cannot be interpreted as linear or circular graphs. Hence, we take a different approach to splicing two graphs and introduce a splicing system for graphs which can be applied to all graphs. Splicing two graphs can be thought of as a new operation among the graphs, that generates new graphs from the given two graphs.

Hence, in this article, the following section discusses the cutting rules, which is the basic component for the proposed graph splicing system. Section 3 deals with the graph splicing system with illustrations. The section 4 studies some graph theoretical properties of this system. The last section concludes with the directions for the future research in this graph splicing system.

2 Definitions

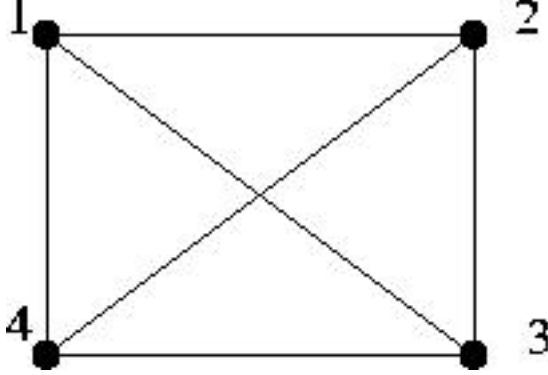
We follow the terminologies and the basic notions of graph theory as in [13] and the terminologies of formal language theory as in [14].

For any finite alphabet Σ , a labeled graph G over V is a triple $G = (V, E, L)$ where V is the finite set of vertices(or nodes), E is finite set of edges of the form (n, m) , $n, m \in V, n \neq m$, where each edge is an unordered pair of vertices and L is a function from V to Σ . An edge (n, m) means that one end-point of the edge is the vertex n and the other end-point is the vertex m . Edge set of G is written as $E(G)$ and the vertex set of G by $V(G)$. The number of vertices of a graph is called the order of the graph and the number of edges of the graph is called the size of the graph. We consider only simple graphs where repeated edges (multiple edges) with same end-points and edges with both end-points same

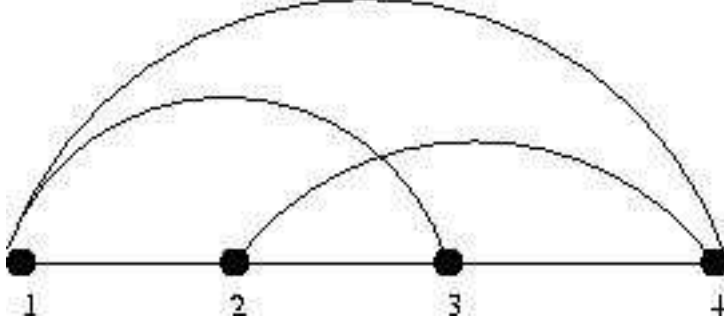
(loops) are not allowed. The graph $G = (V, E, \phi)$ refers to an unlabeled graph. We denote an unlabeled graph just as (V, E) , instead of (V, E, ϕ) . Whenever a graph is considered, we mean only a simple unlabeled graph. We mention accordingly, when we consider the graphs other than the above one.

Definition 1 A graph G is said to be in *Pseudo-Linear Form (PLF)* if the ordered vertices are positioned as per the order, as if they lie along a line and the edges of the graph drawn accordingly.

Ordering of the vertices can be done in any way. For a particular ordering, the adjacency matrix of G and the adjacency matrix of G in PLF, remain the same. In a graph, the vertices could be positioned at any place and the edges of the graph drawn accordingly. For the graph in PLF, vertices are first ordered and positioned as if they lie on a line. This line may be a horizontal line or a vertical line or any inclined line. In case, the line is horizontal, we can position the ordered vertices either from left to right or from right to left. So, without losing any generality, we position the vertices from left to right as if the vertices lie on a horizontal line. Once a graph is in PLF, we name the vertices with a positive integer that represent their order in the ordering. If a vertex is second in an ordering, we name that vertex as 2. So, the vertex set of G in PLF is $\{1, 2, 3 \dots | V | \}$. Given an ordering of the vertices, any graph can be redrawn in the PL form. For example, if the vertices of the graph



are ordered as $\{1, 2, 3, 4\}$, the corresponding graph in PLF is



A graph in PLF will look like a path graph with edges going above or below the linear path. The graph P_n with vertices written horizontally, is a graph in

PLF. From now onwards, unless otherwise mentioned, we mean a graph as the one in PLF for some ordering of the elements of V .

Definition 2 A cutting rule \mathcal{C} for a graph $G = (V, E)$ in PLF is a pair $[i, j]$ ¹, where i and j are positive integers $0 < i \leq j \leq |V|, |j - i| \leq 1$.

By the condition $|j - i| \leq 1$, we mean that the vertices i and j may be successive vertices (ordered successively) or both vertices i and j are the same. A cutting rule $\mathcal{C} = [i, j]$ is called as a reflexive cutting rule if $i = j$.

Definition 3 The left-degree, $ld_G(v)$ of a vertex $v \in V(G)$ is the number of edges of G to the left of the vertex v that are incident with v . The right-degree, $rd_G(v)$ of a vertex $v \in G$ is the number of edges of G to the right of the vertex v . The degree of v , $d(v)$, the number of edges that are incident with v , is the sum of the left-degree and the right-degree of v .

Definition 4 Let $V_l(v)$ be the set of all vertices that lie to the left of the vertex v (v is not in $V_l(v)$). Similarly, $V_r(v)$ is the set of all vertices that lie to the right of the vertex v .

Scheme of cutting

A graph is cut into two parts by cutting some of its edges. The cutting rule $[i, j]$ cuts a graph G between the vertex i and the vertex j (if for some reasons, the vertices are named with symbols other than the positive integers, the cutting rule cuts between the vertex that comes in the i^{th} position in the ordering and with the vertex in the j^{th} position). The work of the cutting rule $[i, j]$ over G is to cut the edge (i, j) and the edges that go above as well as below the edge (i, j) . i.e., The cutting rule $[i, j]$ cuts the following edges (if they exist in the graph G).

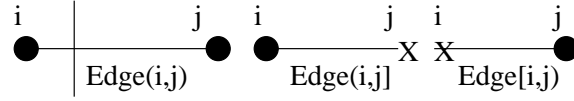
1. The edge (i, j)
2. the edges $(i, v), v \in V_r(j)$
3. The edges $(v, j), v \in V_l(j)$
4. The edges $(u, v), u \in V_l(i), v \in V_r(j)$

The reflexive cutting rule (i, i) cuts the vertex i and all the edges that go above as well as below the vertex i . i.e., the reflexive cutting rule i cuts the following.

1. The vertex i
2. The edges $(u, v), u \in V_l(i), v \in V_r(j)$

¹ for the cutting rule $[i, j]$, we use the square braces and for the edges (i, j) , we use the parenthesis

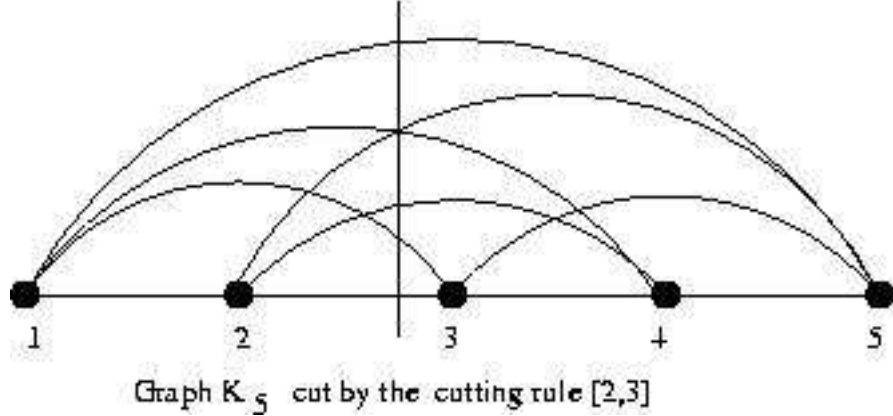
When an edge (i, j) is cut into two parts, we call the the two parts of the edge as hanging-edges or free-edges. Similarly, when a vertex is cut, we call that vertex as a hanging-vertex or a free-vertex. If an edge (i, j) is cut, we write the left part of the edge as $(i, j]$ (indicating that the free-end is the right end) and the right part of the edge as $[i, j)$ (indicating that the free-end is the left end). The edges $(i, j]$ and $[i, j)$ are drawn as illustrated with a \times at their free ends. If a vertex v is cut, the left part of the vertex is written as $v]$ and the right part is written as $[v$. $[v]$ indicates just that the vertex v is cut.

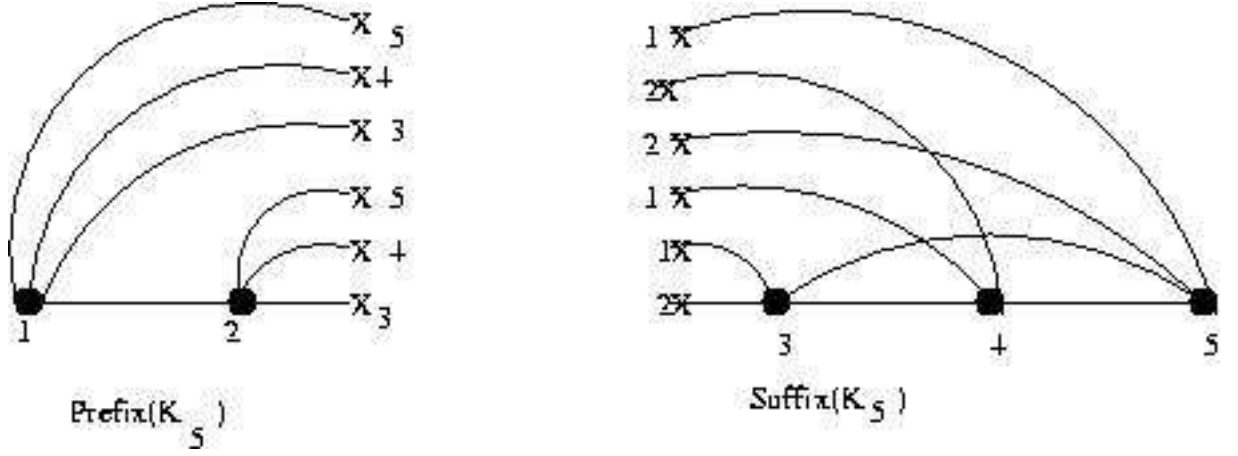


The set $ECUT_G(\mathcal{C})$ represents the set of all edges of G that got cut by the cutting rule \mathcal{C} and the $| ECUT_G(\mathcal{C}) |$ (the cardinality of the set) is the power of the cutting rule \mathcal{C} with respect to the graph G . Power of a cutting rule with respect to G indicates the number of edges that got cut by that cutting rule in G . The set $VCUT_G(\mathcal{C})$ represents the set of all vertices that got cut by the vertex v . Only for the reflexive cutting rules, the set $VCUT_G(\mathcal{C})$ will exist and for all the other cutting rules, this set is ϕ . Since any reflexive cutting rule can cut only one vertex, the set $VCUT_G(\mathcal{C})$ is always singleton. For a reflexive cutting rule, the set $ECUT_G(\mathcal{C})$ can be ϕ (means that no edge is going above or below the vertex i in the graph G).

When a graph G is cut into two by a cutting rule \mathcal{C} , the left part of the graph is called as $Prefix(G)$ and the right part is called as $Suffix(G)$. Obviously, $ECUT_G([i, j]) = ECUT_G([i, i] \cup ECUT_G([j, j] \cup \{(i, j)\})$.

We illustrate the cutting of the graph K_5 , a complete graph with five vertices using the cutting rule $[2, 3]$.





The $prefix(K_5)$ and $Suffix(K_5)$ are also graphs with the vertex set

$$V(Prefix(K_5)) = \{1, 2\}$$

and with the edge set

$$E(Prefix(K_5)) = \{(1, 2), (1, 3], (1, 4], (1, 5], (2, 3], (2, 4], (2, 5]\}.$$

Similarly, $V(Suffix(K_5)) = \{3, 4, 5\}$ and $E(Suffix(K_5)) = \{[2, 3), [2, 4), [2, 5), [1, 3), [1, 4), [1, 5), (3, 4), (3, 5)\}$. $ECUT_{K_5}([2, 3]) = \{(1, 3), (1, 4), (1, 5), (2, 3), (2, 4), (2, 5)\}$. $VCUT_{K_5}([2, 3]) = \phi$.

3 Graph Splicing System

Definition 5 A splicing rule $S = (C_1, C_2)$, is a pair of cutting rules.

Given two graphs G, H and a splicing rule $S = (C_1, C_2)$, the first graph G is cut as specified by C_1 and the second graph H is cut as specified by C_2 . As a result we get the four cut-graphs viz., $Prefix(G), Suffix(G), Prefix(H)$ and $Suffix(H)$.

Mode of recombination

Definition 6 $Prefix(G)$ (or $Prefix(H)$) recombines with the $Suffix(H)$ (or $Suffix(G)$) if and only if $|ECUT_G(C_1)| = |ECUT_H(C_2)|$ and $|VCUT_G(C_1)| = |VCUT_H(C_2)|$. In other words, for a recombination, the number of hanging-edges in $Prefix(G)$ (or $Prefix(H)$) should be the same as that of the number of hanging-edges in $Suffix(H)$ (or $Suffix(G)$) and the number of hanging-vertices in $Prefix(G)$ (or $Prefix(H)$) should be the same as that of the number of hanging-vertices in $Suffix(H)$ (or $Suffix(G)$).

The above definition tells that for a splicing process to end up in a recombination, the power of both the cutting rules present in the splicing rule \mathcal{S} should be the same. We assign a positive integer, called the power of the splicing rule, to every splicing rule $\mathcal{S} = (\mathcal{C}_1, \mathcal{C}_2)$ if and only if the powers of \mathcal{C}_1 and \mathcal{C}_2 are the same and the common value is the power of the splicing rule \mathcal{S} . Further, if one cutting rule in \mathcal{S} is reflexive, the other should also be reflexive.

Definition 7 *Every hanging-edge of the $Prefix(G)$ (or $Prefix(H)$) recombines (or joins) with only one hanging-edge of the $Suffix(H)$ (or $Suffix(G)$), and every hanging-edge of $Suffix(H)$ (or $Suffix(G)$) has the recombination with only one hanging-edge of $Prefix(G)$ (or $Prefix(H)$). The hanging-vertex (if available) of the $Prefix(G)$ (or $Prefix(H)$) recombines with the hanging-vertex of $Suffix(H)$ (or $Suffix(G)$).*

Thus, $Prefix(G)$ recombines with $Suffix(H)$ to generate new graphs. After the recombination, we order the vertices of the new graph (this will be in PL form) in the same sequence as it appears and name them accordingly. New graphs are generated because of the recombination of the edges that are cut. If there are more than one hanging-edges in both $Prefix(G)$ and $Suffix(H)$, the hanging-edges of the $Prefix(G)$ can recombine with the hanging-edges of $Suffix(H)$ in more than one way. If there are m hanging-edges in both $Prefix(G)$ and $Suffix(H)$, the hanging-edges can recombine in $m!$ ways, generating $m!$ new graphs. In other words, the number of such recombinations is just the number of bijective mappings from the set $ECUT_G(\mathcal{C}_1)$ to the set $ECUT_H(\mathcal{C}_2)$. When the $Prefix(H)$ recombines with the $Suffix(G)$, the same number of $m!$ will be generated. Thus, the splicing of two graphs G and H using a splicing rule of order m , generates $2(m!)$ new graphs.

Thus, splicing process comprises of cutting as well as the recombination. If the splicing of G and H using \mathcal{S} generates a new graph F by the recombination of the $Prefix(G)$ with the $Suffix(H)$, we denote that by $G \vdash_{\mathcal{S}}^1 H = F$ (indicating that F is the first splicing product). Similarly, $G \vdash_{\mathcal{S}}^2 H = F$ indicates that F is generated by the recombination of the $Prefix(H)$ with the $Suffix(G)$ (indicating that this F is the second product of splicing). Just $G \vdash_{\mathcal{S}} H = F$ indicates that F may be either the first splicing product or the second splicing product. The splicing scheme (process) is denoted by σ . For a splicing process, one requires two graphs and a splicing rule. The set of all graphs generated by splicing G and H using the splicing rule \mathcal{S} is denoted by $\sigma(\{G, H\}, \mathcal{S})$. Similarly, $\sigma_1(\{G, H\}, \mathcal{S})$, $\sigma_2(\{G, H\}, \mathcal{S})$ are meant accordingly.

Definition 8 *The Graph Splicing System $\nu = (\mathfrak{A}, \mathfrak{S})$, where*

\mathfrak{A} *A finite set of simple, unlabeled graphs, called the set of axioms.*

\mathfrak{S} *A finite set of splicing rules.*

The underlying splicing scheme is $\sigma(\{G, H\}, \mathcal{S})$, $G, H \in \mathfrak{A}, \mathcal{S} \in \mathfrak{S}$.

The set of all graphs generated by splicing all pairs of the graphs of \mathfrak{A} with all

splicing rules of \mathfrak{S} (The graph language of the splicing system ν),

$$L(\nu) = \sigma(\mathfrak{A}) = \bigcup_{G, H \in \mathfrak{A}, S \in \mathfrak{S}} \sigma(\{G, H\}, S)$$

In the DNA recombination, when some restriction enzymes and a ligase are present in a test tube, they do not stop after one cut and paste operation, but they act iteratively. The products of a splicing again take part in the splicing process. For an iterative splicing among the graphs, the axiom set should contain many copies of the same element. Ordinary sets are composed of pairwise different elements, i.e., no two elements are the same. If we relax this condition, i.e., if we allow multiple but finite occurrences of any element, we get a generalization of the notion of a set which is called a *multiset*. We assume that our axiom set is a multiset. That means infinitely many copies of the elements of the axiom set will be present in the set, which facilitates the elements to take part in the splicing process iteratively. Even the product of a splicing process will also be available infinite number of times. To make a graph splicing system into an iterated graph splicing system, the only requirement is to make the axiom set \mathfrak{A} into a multiset such that infinitely many copies of the elements of \mathfrak{A} are in \mathfrak{A} .

Definition 9 The graph language of an iterative graph splicing system $\nu = (\mathfrak{A}, \mathfrak{S})$, where \mathfrak{A} is a multiset such that infinitely many copies of the elements of \mathfrak{A} are in \mathfrak{A} , is defined as $L(\nu) = \sigma^*(\mathfrak{A})$ where

$$\begin{aligned} \sigma^0(\mathfrak{A}) &= \mathfrak{A}, \\ \sigma^{i+1}(\mathfrak{A}) &= \sigma^i(\mathfrak{A}) \cup \sigma(\sigma^i(\mathfrak{A})), \\ \sigma^*(\mathfrak{A}) &= \bigcup_{i \geq 0} \sigma^i(\mathfrak{A}). \end{aligned}$$

Example 1 Consider the graph splicing system $\nu = (\{C_3, C_4\}, \{([1, 2], [2, 3])\})$. C_3 and C_4 are the cycle graphs of order 3 and 4 respectively.

$$L(\nu) = \sigma(\{C_3, C_4\}, \mathcal{S}) \cup \sigma(\{C_4, C_3\}, \mathcal{S}) \cup \sigma(\{C_3, C_3\}, \mathcal{S}) \cup \sigma(\{C_4, C_4\}, \mathcal{S})$$

where \mathcal{S} is the splicing rule $([1, 2], [2, 3])$. The power of the splicing rule is 2. In each splicing process, $2(2!) = 4$ new graphs will be generated. So, $L(\nu)$ will have a total of 16 new graphs. Of these, some of the graphs are isomorphic to each other. It is found that the non-isomorphic graphs in $L(\nu)$ are C_3, C_4, C_5 and a graph G , where $G = (\{1, 2\}, \{(1, 2), (1, 2)\})$. The above graph G is not a simple graph (it has a multiple edge between the vertices 1 and 2). This makes us to conclude that the splicing of two simple graphs need not be simple.

4 Properties

Proposition 1 *Given a graph G , the power of the cutting rule $[i, j]$ with respect to the graph G is*

$$rd(i) - ld(i) + \sum_{v \in V_l(i)} (rd(v) - ld(v)).$$

Proof: Let G be the given graph. We count the total the number of edges in G that got cut by the cutting rule, which is the power of the cutting rule. We classify the proof into two cases based on the existence of the edge (i, j) in G or not.

Case(i) : $(i, j) \in E(G)$.

We know that the cutting rule $[i, j]$ cuts the following edges.

1. The edge (i, j)
2. The edges $(i, v), v \in V_r(j)$
3. The edges $(v, j), v \in V_l(j)$
4. The edges $(u, v), u \in V_l(i), v \in V_r(j)$

The expression

$$rd(i) + ld(j) - 1 \tag{1}$$

brings out the number of edges which fall under (1),(2) and (3) in the list above. Since the edge (i, j) is counted in both $rd(i)$ as well as in $ld(j)$, we subtract one from the expression.

Let A be the set of edges whose left end is $V_l(i)$. Let $B \subset A$, be the set of edges of A whose right end is in $V_l(i)$. i.e., both the ends of edges in B are in $V_l(i)$. Let $C \subset A$ be the set of edges of A whose right end is i i.e., for the edges in C , one end is $V_l(i)$ and the other end is i . Let $D \subset A$, be the set of edges of A whose right end is j . i.e., for the edges in D one end is in $V_l(i)$ and the other end is j . Let E be the set of edges of A whose right end is in $V_r(j)$ i.e., the set of edges whose left end is in $V_l(i)$ and the right end is in $V_r(j)$ Obviously, the set of edges which come in (4) in the above list, will be E .

$$|A| = \sum_{v \in V_l(i)} rd(v); |B| = \sum_{v \in V_l(i)} ld(v); |C| = ld(i); |D| = ld(j) - 1.$$

Since the edge (i, j) would be counted in $ld(j)$, we subtract one from $ld(j)$.

Number of edges that come under (4) is

$$|E| = |A| - |B| - |C| - |D| = \sum_{v \in V_l(i)} rd(v) - \sum_{v \in V_l(i)} ld(v) - ld(i) - (ld(j) - 1) \tag{2}$$

Hence, the total number of edges cut by $[i, j]$

$$= (1) + (2) = rd(i) - ld(i) + \sum_{v \in V_l(i)} (rd(v) - ld(v))$$

Case(ii) : (i, j) not in $E(G)$

We proceed similarly as the case(i). The number of edges that come under (1),(2) and (3) is

$$rd(i) + ld(j)$$

Number of edges that come under (4) is

$$= \sum_{v \in V_l(i)} rd(v) - \sum_{v \in V_l(i)} ld(v) - ld(i) - ld(j)$$

Hence, the total number of edges cut by $[i, j]$

$$= rd(i) - ld(i) + \sum_{v \in V_l(i)} (rd(v) - ld(v))$$

In both the cases, we get the same expression. Hence the proof.

Theorem 1 *In any graph G , the sum of the differences between the right degree and the left degree of all the vertices is zero.*

Proof: In the Proposition 1, in computing the power of a cutting rule $[i, j]$, we counted the number of edges whose one end is in $V_l(i)$ and the other end is in $V_r(j)$ by deleting some edges from the set A which is the set of edges whose left end is in $V_l(i)$. Instead, we can have the set A to be the set of edges whose right end is in $V_r(j)$ and proceed in an analogous way, as in the proof of Proposition 1. We get the power of the cutting rule $[i, j]$ to be

$$ld(j) - rd(j) + \sum_{v \in V_r(j)} ld(v) - rd(v)$$

which is a symmetric one with the expression got in Proposition 1.

Since the power of a cutting rule is a constant with respect to a G , both the expressions should be equal.

$$rd(i) - ld(i) + \sum_{v \in V_l(i)} (rd(v) - ld(v)) = ld(j) - rd(j) + \sum_{v \in V_r(j)} (ld(v) - rd(v))$$

$$\text{implies } \sum_{v \in V} (rd(v) - ld(v)) = 0 \text{ or } \sum_{v \in V} (rd(v) - ld(v)) = 0$$

corollary 1 *The number of edges in a graph G is always*

$$\sum_{v \in V} rd(v)$$

or

$$\sum_{v \in V} ld(v)$$

Proof

$$\sum d(v) = \sum (ld(v) + rd(v)) = 2 | E |$$

we have

$$\sum (ld(v) - rd(v)) = 0.$$

This implies,

$$\sum ld(v) = | E | = \sum rd(v)$$

Remark 1 *The above Corollary can also be proved in another way using the fact that every edge should contribute one to the left degree of some vertex and one to the right degree of some other vertex.*

For want of space, We state some of the results without proofs.

Theorem 2 1. $G \vdash H = H \vdash_{S^R} G$, where S^R is the splicing rule in which the cutting rules of S got swapped.

2. $G \vdash_S H \neq H \vdash_S G$ i.e., the splicing operation is not commutative.

3. The splicing operation preserves the degrees of the vertices

4. Regularity is preserved by the splicing. i.e., if we splice any two regular graphs, the splicing product is again a regular graph.

5. maximum size of the splicing product of G and H will be the sum of the orders of G and H minus 1.

6. For a complete graph K_n , $rd(i) = ld(n+1-i)$, for every vertex $i \in V(K_n)$.

7. The set of all simple graphs is not closed with respect to the splicing operation.

Theorem 3 A graph G is said to contain a cycle if and only if there exists a sequence A of successive cutting rules ² with power > 1 such that

$$\bigcap_{[i, i+1] \in A} ECUT_G([i, i+1]) \neq \phi$$

²The rules $[i, i+1]$ and $[i+1, i+2]$ are termed successive cutting rules

Theorem 4 *Let G and H be any two isomorphic graphs. Let $G \vdash_{\mathcal{S}} H = F$, for any splicing rule \mathcal{S} . Then F is isomorphic to G (or H) if and only if the order of the graph F and the order of G (or the order of H) are the same.*

Theorem 5 *If for a graph G , there exists only one cutting rule whose power is equal to the size of the graph G , Then G is bipartite.*

5 conclusion

As graphs are better suited for representing complex structures, a model for splicing the graphs, graph splicing system is introduced, which can be applied to all types of graphs. Though the graph splicing is introduced as a new operation among the graphs, studying the computational effectiveness of this graph splicing system is an important area to explore. One can introduce various parameters like the number of graphs in the axiom, the number of splicing rules, power of the splicing rule etc., and finding the minimum value of the parameters for which the graph splicing system is still computationally complete. Besides, as a new line of thinking, a nice investigation to bring out the utility of the splicing in graph theory is worth.

References

- [1] Searls,D.B., 1992, The linguistics of DNA, *American Scientist*, **80**, 579-591.
- [2] Colaldo-vides,J., 1991, The search for a grammatical theory of gene regulation is formally justified by showing the inadequacy of context-free grammars, *CABIOS*, **7**, 321-336.
- [3] Head,T., 1987, Formal language theory and DNA : An analysis of the generative capacity of specific recombinant behaviours, *Bull.Math.Biology*, **49**, 737-759.
- [4] Paun,Gh., 1996, On the Splicing operation, *Discrete Applied Mathematics*,**70**, 57-79.
- [5] Paun,Gh., Rozenberg,G and Salomaa,A., 1996, Computing by Splicing, *Theoretical Computer Science*, **168**(2), 321-336.
- [6] Freund,R., 1995, Splicing systems on graphs, *IEEE conf. on Intelligence in Neural Biological systems, Herndon-Washington*, 189-195.
- [7] Culik II,K., and Harju,T., 1991, Splicing semigroups of dominoes and DNA, *Discrete Appl.math*, **31**, 261-277.
- [8] Rama, R. and Umaraghavan,Splicing Array Systems, *Intern. J. Computer. Math.*,, **73**,167-182.

- [9] Rama, R. and Krishna, S.N., 1999, Contextual Array Splicing systems, *Proceedings SPIRE'99 & CRIWG'99*, 168-175.
- [10] Krithivasan, K., Chakravarthy, V.T. and Rama, R., 1997, Array Splicing Systems, LNCS 1218, 346-365.
- [11] Kari,L., 2001, DNA computing in vitro and in vivo, *Futute Generation Computer Systems*,**17**, 823-834
- [12] Salomaa,A., 1997, Computability paradigms based on DNA complementarity, *in V.Keranen(ed), Innovation in Mathematics, Proc.,second International Mathematics Symposium, Computational Mechanics Publications, Southampton and Boston*,15-28.
- [13] Bondy, J.A and Murty, U.S.R., 1976, Graph Theory with Applications *North-Holland, New York*.
- [14] Hopcroft, J.E and Ullman, J.D., 1979, Introduction to Automata Theory, Languages and Computation, *Addition-Wesley,R Reading, Mass.,*.
- [15] Santhanam,R.,and Krithivasan,K., 2006, Graph Splicing Systems, *Discrete Applied Mathematics*, **154**,1264-1278.